APPLYING QUALITY OF SERVICE ARCHITECTURES TO THE FIELD-BUS DOMAIN [1]

Frank Rößler[*] and Birgit Geppert[**]

Computer Science Department, University of Kaiserslautern, P.O. Box 3049
67653 Kaiserslautern (Germany), Fax: +49 631/205-3956
* Phone: +49 631/205-3272, Email: roessler@informatik.uni-kl.de
** Phone: +49 631/205-2623, Email: geppert@informatik.uni-kl.de

Abstract - *Communication between intelligent sensors, actuators and controller components imposes stringent requirements on timeliness and predictability. Thus field-bus standards such as PROFIBUS, EIB, or CAN support deterministic, priority-driven media access. These mechanisms are sufficient for static settings where communicating peers as well as traffic loads are known in advance and priorities can be assigned prior to system operation. In environments where, e.g., communicating devices may change dynamically we need to provide additional mechanisms for admission control and traffic policing, which operate during system activity. Local and Wide area networks have to meet similar requirements, which led to the development of so-called quality of service architectures. We argue that many of these concepts are also adequate to the field-bus domain and adapt them to the CAN protocols. As a basic means to achieve timeliness and predictability we employ rate monotonic analysis.*

1. INTRODUCTION

To support timeliness and predictability field-bus standards such as Process Field Bus (PROFIBUS) [1] [6], European Installation Bus (EIB) [10], or Controller Area Network (CAN) [7] [3] provide mechanisms for deterministic and priority-driven (statically assigned) media access.

PROFIBUS proposes a token-passing protocol between master stations where the time each master may continuously send messages is bounded by a token holding time. The token holding time results from the difference between a predefined target-rotation-time and the actually observed real-rotation-time of the token. Additionally two message priority levels are defined. Low priority messages are sent, if there are no high priority messages in the output queue. As far as high priority message transfers are pending, a master station is guaranteed to transmit one high priority message per received token (no matter if there is enough token holding time left). In this way PROFIBUS guarantees that successive high priority messages will leave a particular master node with a bounded time span between. Note that this only guarantees a maximum transmission delay for the first high priority message in the output queue. To calculate the maximum time spent inside the output queue we need additional knowledge about the traffic load at all master stations. Furthermore the target-rotation-time must be configured accordingly.

On the other hand EIB and CAN employ a bitwise bus arbitration based on control/address-fields and message identifiers, respectively, to provide for a deterministic media access. Control/address-fields and message identifiers are to be interpreted as priorities and therefore a global, priority-driven scheduling of message transfers is realized. As a consequence maximum transmission delays depend on the total network load.

As timeliness and predictability strongly depends on worst case traffic load, field-bus installation normally precedes a configuration step where involved communicating peers as well as traffic loads are identified and priorities are assigned accordingly. This step is carried out prior to system operation and works fine for static settings. Nevertheless the operator or owner of a field-bus installation may change communicating devices or applications dynamically (e.g., in building automation). Though dynamic insertion and removal of communicating devices is generally supported by field-bus standards, timeliness guarantees have to be concerned from scratch.

It would be an improvement to perform admission control, priority assignment, and traffic policing *automatically during system operation*. This may be realized as a kind of "plug and play" functionality for timeliness guarantees, i.e. when, for instance, a new communicating device is inserted, the system should automatically handle priority assignment with respect to timeliness requirements and under consideration of the already existing traffic load. If there are not enough network

1. This work has been partially supported by the German Science Foundation (DFG) as part of the Sonderforschungsbereich SFB 501 *Development of large systems with generic methods*

Proceedings of the 2nd IEEE International Workshop on Factory Communication Systems, WFCS'97, Barcelona, Spain, 1997

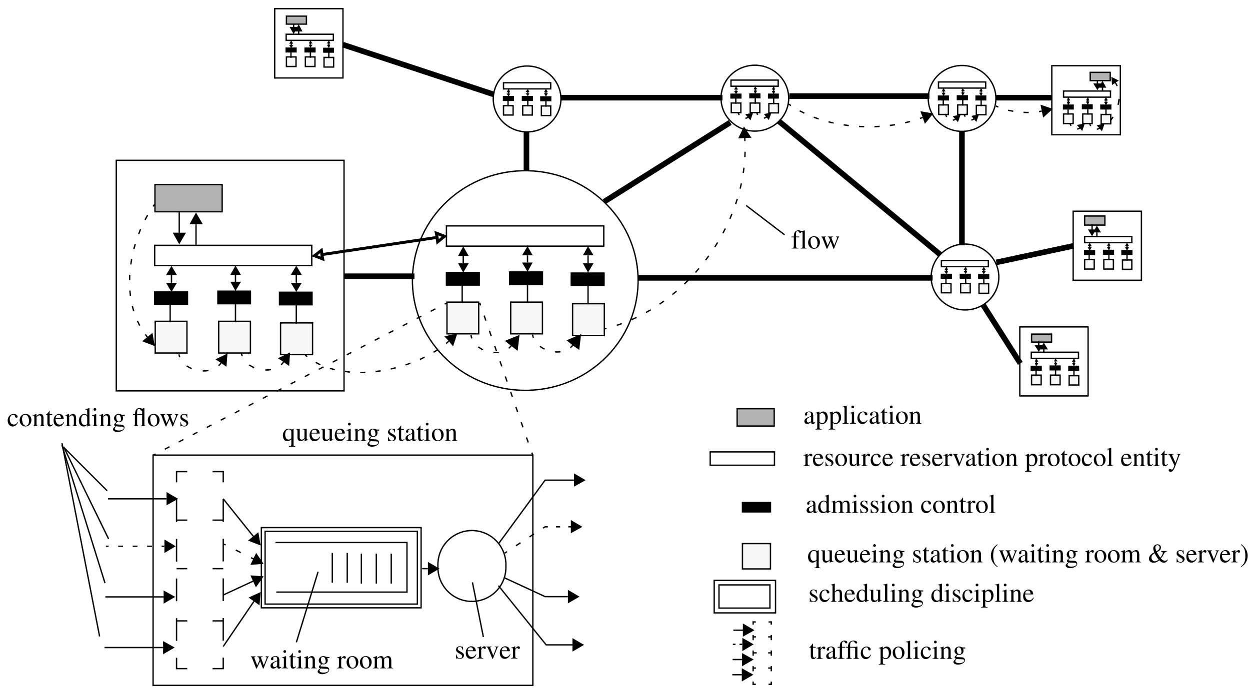

Fig. 1: generic QoS-architecture

resources left, the system may deny additional communicating peers or propose a particular reconfiguration.

Due to quality of service (QoS) requirements of modern applications Local and Wide area networks have to meet similar requirements, which led to the development of so-called QoS architectures. We argue that many of these concepts are also applicable to the fieldbus domain.

The remainder of the paper is organized as follows: In Section 2 we introduce the main principles of current QoS architectures and relate them to the mechanisms provided by PROFIBUS, EIB, and CAN. Section 3 describes more detailed features of CAN, which serve as a basis for QoS relevant extensions. Finally, Section 4 presents an instantiation of a QoS architecture for CAN, i.e. actual protocol extensions for the CAN application layer, which implement the above mentioned plug and play functionality with respect to timeliness guarantees. We conclude with a summary in Section 5.

2. PRINCIPLES OF CURRENT QOS ARCHITECTURES

Due to evolving high-speed transmission and network technologies (e.g., fibre optics and ATM) it is basically possible to meet ever-increasing user requirements on throughput and end-to-end delay. As a consequence traditional network services such as remote login or file transfer nowadays have enough network resources available to perform properly without any kind of resource management. Modern applications (e.g., remote visualization or virtual reality), however, are more demanding w.r.t. bandwidth, delay, or delay jitter and will dramatically degrade in performance when resource sharing is not suitably restricted. This section introduces the main concepts found in current QoS architectures in order to meet stringent user requirements on throughput, delay, delay-jitter, and reliability.

2.1 Network model

For the following discussion we assume a packet-switched network where the nodes are connected by point-to-point links. Figure 1 gives an example with end systems and intermediate systems depicted as rectangles and circles, respectively.

On its way from the sending application to the receiver side each data packet will be processed by a sequence of active resources such as CPUs or network adapters. In Figure 1 active resources are represented as queueing stations (shaded ☐) consisting of one server and an associated waiting room. Queueing stations either reside on end systems or intermediate systems. Generally data packets from different end-to-end connections (flows) contend for the same network resources and thus mutually influence delivered quality of service. On principle any QoS architecture aims to separate mutual interaction of different flows w.r.t. quality of service.

2.2 Service model

Contrary to the common best-effort service model (provided, for instance, by the Internet) QoS architectures support data transfer with *guaranteed* quality of service [8]. That is the service user and service provider are contracting parties where the service provider commits to transmit user data with certain QoS guarantees, if the service user does not exceed a particular traffic load.

Note that QoS parameters and traffic load are negotiated between the contracting parties during connection establishment.

Common QoS parameters are end-to-end delay, delay-jitter, and packet loss where obligation of guarantees is handled differently by existing approaches. On the one hand there are deterministic and statistical guarantees [9]. On the other hand some approaches differentiate between guaranteed service (delivered QoS only depends on the clients commitment not to exceed negotiated traffic load) and predicted service (delivered QoS additionally depends on the behaviour of other network clients, i.e. QoS guarantees are not based on worst-case assumptions) [4]. The reason why several service commitments are in use is that service providers find different trade-offs between efficient usage of network resources and obligation of guarantees.

Note that for specification of traffic loads also various traffic models are proposed in the literature.

2.3 QoS provision

Data packets have to traverse a sequence of queueing stations to finally reach their destination where the local behaviour of each queueing station contributes to the globally delivered QoS. As a matter of fact current QoS architectures rely on a connection-oriented approach, i.e. all data packets of a flow follow the same route. During connection establishment this route is fixed (virtual connection) and for each traversed queueing station (see Figure 1) a certain amount of capacity (e.g., CPU time, bandwidth, or buffer) is reserved. As a consequence data packets can be processed with predictable performance during data transfer. For preparation of a suitable route basically two mechanisms are necessary: a resource reservation protocol and an admission control test (shaded and in Figure 1) which are introduced in the following.

Resource reservation protocol

A resource reservation protocol establishes a virtual connection and distributes QoS relevant information to the traversed network nodes. At the same time reservation of sufficient resources is usually initiated in order to guarantee best possible QoS on each network node. When the route has been completed surplus resources may be released subsequently.

Under assumption of unidirectional and unicast communication a connection is established by a two-way handshake. While the connect-request-packet traverses the network (according to existing routing tables), routing information is stored in the visited nodes and -in addition- reservations for best possible QoS are initiated. Accumulated QoS of the so far established path is sent along with the connect-request-packet to the next hop. At the receiver side it comes out whether the selected route offers suitable QoS to meet the user requirements. Accordingly either a connect-accept or connect-refuse packet is sent back on the same route (but in the reverse direction) and reserved resources are either relaxed (in the case of surplus reservations) or completely released. Example resource reservation protocols are RSVP [25] and ST2+ [5].

Admission control

Admission control provides the mechanisms to reserve, relax, or release resources and to test the availability of sufficient resources to meet user requirements. With each queueing station such an interface to the resource reservation protocol is associated as shown in Figure 1. Admission control also compares actual end-to-end QoS of a particular route against the user requirements and decides whether resources are to be relaxed. Note that admission control may only accept a new connection, if QoS guarantees of already existing connections are not violated. It is important mentioning that actual admission control algorithms highly depend on the nature of the underlying resource and the QoS control mechanisms applied (see [14] for an example).

2.4 QoS control

QoS provision exclusively operates during connection establishment and controls total network load by accepting or refusing further connections. QoS control, however, operates on packet level during data transfer and controls traffic load of individual queueing stations. Similar to daily life (imagine you have reserved a hotel room but somebody else got it just before you arrived) reservations are useless, if nobody enforces the intended usage of resources. Generally there are two QoS control mechanisms: traffic policing (shaded in Figure 1) which prevents individual flows from overloading the resources (i.e. consuming more resources than agreed on during connection establishment) and the scheduling discipline (shaded in Figure 1) which regulates packet processing orders to impose individual waiting times on different flows.

Traffic policing

Traffic policing ensures that each flow behaves as contracted, because the packet arrival pattern at each queueing station must comply the expected (from the scheduler's viewpoint) traffic load. Packets of a particular flow that arrive early are delayed (rate-jitter control [26]), dismissed, or otherwise punished (under certain circumstances some scheduling disciplines allow processing of early packets without violating other performance guarantees [9], [24]). As a consequence traffic policing ensures throughput guarantees and - in addition - protects the scheduler from overload situations (even transient). Sometimes traffic policing algorithms compensate for delay-jitter by completely reconstructing the original traffic pattern (as observed at the network entry points) on entry into the queueing stations [23]. Note that violation of user specified traffic load may either happen because of misbehaving senders or varying network load which distorts traffic patterns.

Scheduling discipline

The scheduling discipline resolves contention of the different flows for the particular server (e.g., CPUs or network interface). As the scheduler orders the processing of the packets from all the connections it imposes individual maximum waiting times on the different flows. For this purpose various scheduling disciplines have been conceived, for instance, rate-monotonic-scheduling [20] [17], earliest-deadline-first [20], virtual-clock [24], weighted fair queueing [4] [21] [22], stop-and-go [11] [27], or hierarchical round robin [15]. As we are concerned with real-time communication a scheduling algorithm may not be too complex (concerning run time) and must allow to calculate upper bounds on waiting times.

2.5 Further issues

The cursory introduction to resource management given in Section 2 is sufficient for our purpose but by no means complete. We left many other interesting issues out. For instance, we did not mention multicast support, dynamic adaptation of QoS, QoS mapping, or other QoS characteristics such as connection setup delay. Additionally we only concerned so-called open-loop approaches and ignored the field of closed-loop control with related issues such as QoS monitoring, QoS maintenance, and QoS scalability (see [2] for a more comprehensive overview on these topics).

2.6 Relation to field-bus standards

Sections 2.3 - 2.4 have identified four essential components of a generic QoS architecture: resource reservation protocol, admission control, traffic policing, and scheduling discipline. An examination of the field-bus standards PROFIBUS, EIB, and CAN yields that apart from the scheduling discipline all functionalities are missing. Note that CAN supports traffic policing by blocking successive message transmissions for a certain time span. This, however, will be discussed in the next section. As already mentioned in Section 1 resource reservation protocols, admission control, and traffic policing are indeed superfluous in static settings where communicating peers and traffic loads are known in advance and priorities are assigned prior to system operation. This is true because of the following reasons.

- Admission control and resource reservation (i.e., assignment of priorities and token holding times) are handled manually during system configuration. Thus there is also no need for a resource reservation protocol.
- Traffic sources are expected to behave exactly as specified. Thus traffic policing at the network entry points is not necessary.
- A field-bus is a broadcast medium and not a typical store-and-forward network. Thus there are no intermediate systems and therefore no need for traffic policing inside the network.

The situation changes when we allow communicating peers to be created and destroyed dynamically during system operation. Then we are either forced to reconfigure the system manually and possibly disrupt system activity or we extend system functionality by at least an admission control mechanism and a resource reservation protocol. As there are no intermediate systems the resource reservation protocol can be simplified (Section 4.5). By incorporating QoS mechanisms we realize a kind of plug and play functionality for timeliness guarantees. Only if there are not enough network resources left, the admission of a new connection will be refused or a reconfiguration of network resources is initiated. Even if we can rely on traffic sources, which behave exactly as specified, traffic policing may also be useful to increase system robustness.

So far we identified functional units of a generic QoS architecture but did not provide any possible implementations. In Section 4 we propose an actual instantiation of the generic QoS architecture for CAN which realizes the above mentioned plug and play functionality.

3. CONTROLLER AREA NETWORK (CAN)

From the three field-bus standards considered during our work we have chosen CAN to be further investigated w.r.t. resource management. The point is that the CAN medium access control is best suited for timing analysis using scheduling theory (Section 4.2). In the following we discuss the features of CAN relevant to QoS where Sections 3.1, 3.2 describe data transfer and medium access as part of the ISO specified data link layer [12] [13] and Section 3.3 introduces the CAN application layer CAL specified by CAN-in-Automation e.V. (CiA) [7].

3.1 Data transfer

In CAN data transfer is based on broadcast communication, i.e. each frame is basically offered to all communicating entities. However, receivers make a selection based on an identifier, which is carried in the header of each (data/remote) frame[1]. Actually these identifiers describe the type of the transmitted data object. As object identifiers are also employed for medium access control (Section 3.2) the same identifier may only be used by one sender. CAN defines four frame types:

- With a *data frame* a sender offers a piece of data to all the other communicating entities. It carries an object identifier for selection on the receiver side.

1. Note that CAN does not support sender or receiver addressing.

- With a *remote frame* a receiver may request the offering of a certain object. Note that the receiver is not aware of the location of the data source. However, the remote frame carries the proper object identifier in its header and the corresponding sender will recognize it.
- A communicating device signals error situations with an *error frame*. All other communicating entities will process this frame.
- With an *overload frame* a receiver may temporarily delay transmission on the medium.

In normal operation (i.e., if no errors occur) CAN realizes a connectionless multicast communication where receivers may join multicast groups freely (they only need to know the corresponding object identifiers). In addition receivers may independently request transmission of particular objects by the use of remote frames.

SOF(1)	Object Identifier(11)	RTR(1)	Control(6)	Data(0..64)	CRC(16)	ack(2)	EOF(1)

Fig. 2: data/remote frame format

Figure 2 shows the frame format of data and remote frames where numbers in parentheses denote the field length in bit. The frames start and end with a Start-of-Frame (SOF) bit and an End-of-Frame (EOF) bit, respectively. The Remote-Transmission-Request (RTR) bit differentiates between data and remote frames. In the case of a data frame the Object Identifier denotes the transmitted object. In the case of a remote frame the Object Identifier denotes the requested object. The Data field may be up to 8 bytes long (however, no data is transmitted by a remote frame) where the actual length in bytes is given by the Control field (two bits of the Control field, however, are reserved for future purposes). Furthermore CAN defines various error control mechanisms (e.g., bit monitoring, message frame check, CRC, acknowledgements) which also use the CRC and Acknowledgement fields. Detected error situations are immediately signalled by the means of an error frame. We ignore the actual format of an error frame, which basically consists of a sequence of dominant signal levels that violates certain coding rules as well as the format of an overload frame.

3.2 Medium access control

As CAN relies on a shared medium it is principally possible that several communicating devices try to access the medium at the same time. As a suitable access control CAN employs carrier sense multiple access with collision avoidance (CSMA/CA). That is communicating devices may not jam ongoing transmissions (carrier sense) and simultaneous starts of frames are resolved without disturbing the frame of the winning sender (collision avoidance). Collision avoidance is realized by a bitwise arbitration during transmission of the Object-Identifier and the RTR field. Each sender monitors the current medium signal level and terminates its own transmission, when the medium signal level and the own send signal level differ. This may only happen (in the absence of electrical inference), if two senders transmit simultaneously. In order to choose a winner and not to disrupt the winner's frame CAN implements signal levels either dominant or recessive, i.e. simultaneous transmission of a dominant and a recessive signal keeps the dominant signal level and absorbs the recessive. As a consequence the sender that transmits the first recessive signal during the arbitration phase will lose and terminate. As already mentioned object identifiers are globally unique and therefore it is guaranteed that a collision will always be successfully resolved. However, it is possible that the same object identifier may be sent simultaneously in a data and remote frame. The RTR bit of a remote frame is by definition recessive (and dominant in a data frame) and therefore the transmission of a particular object precedes a simultaneous request. Figure 3 shows an example arbitration phase with three contending senders. By definition the Start-of-Frame bit is dominant. It can be seen that sender 2 transmits a data frame and loses after transmission of seven bits. Sender 1 and sender 3 carry the same object identifier. This is possible, because sender 3 only transmits a remote frame. As a result sender 1 gains exclusive medium access at the end of the arbitration phase and transmits the rest of the frame subsequently.

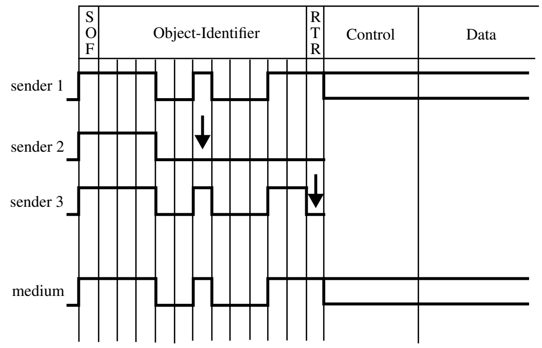

Fig. 3: bitwise arbitration

3.3 CAN application layer CAL

The CAN application layer corresponds to layer 7 of the ISO/OSI reference model and has been standardized by the CAN-in-Automation users and developers group. CAL provides services for network management (initialization, configuration, and monitoring of communicating devices), assignment of object identifiers, and local configuration of the communication subsystem. Additionally CAL defines communication services for the remote access to process variables and the distribution of events. In the following we will discuss assignment of object identifiers as well as some communication services in more detail.

Communication services

From a client's point of view CAL offers a number of objects, which represent remote process variables or remote process events. The client may read or write variables and is notified, whenever an event has happened. From a server's point of view CAL offers a number of objects, which represent local process variables and events, which occur locally. The server is responsible to keep conformance between the real process and the process image described by its objects. A client and a server are connected, if they share a common object. Note that an object may have several clients or servers. This depends on the semantics of the actual object. Additionally an application process may play the role of several servers and clients but for separate objects.

CAL offers different object types. Basic types are read-only variable, write-only variable, and uncontrolled event. The semantics of a read-only variable is illustrated in Figure 4. If a client invokes a read request on a particular read-only variable, CAL will make the data link layer to request for the corresponding object (by the means of a remote frame[1]). As a consequence the remote data link entity responds with the current value of the requested object. Finally, CAL hands the value out to the client. Note that the server is responsible to periodically update the object's contents (via an update request) and that the server gets a read indication whenever the object is accessed. A read-only variable needs exactly one server and may have several clients.

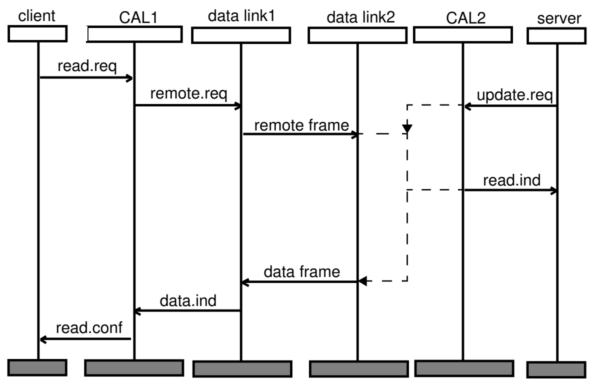

Fig. 4: confirmed service

As another example Figure 5 shows the semantics of an uncontrolled event. If an event occurs, the corresponding server can report this by a notify request to all the clients that share the particular object.

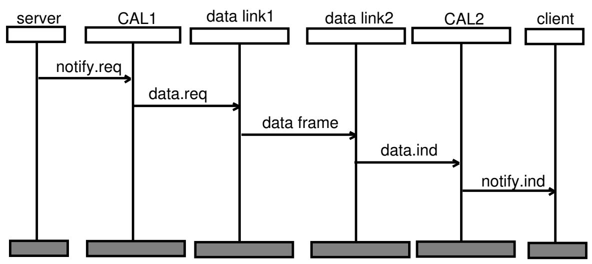

Fig. 5: unconfirmed service

1. Note that CAL objects correspond directly to data link layer object identifiers (Figure 2).

CAL additionally provides for some enhanced objects, for instance, read-write variables, controlled events, and domains. Thereby a controlled event may be masked by a client, so that a notify request will not reach the clients. Domains allow for the transfer of data that is more than eight bytes long (e.g., program or parameter downloads).

Assignment of object identifiers

CAL objects correspond directly to object identifiers, i.e. for each CAL object there has to be assigned an unique object identifier that is to be used for addressing purposes on the data link layer. CAL provides for a special service (distributor, DBT) to assign object identifiers during system initialization. Figure 6 illustrates the

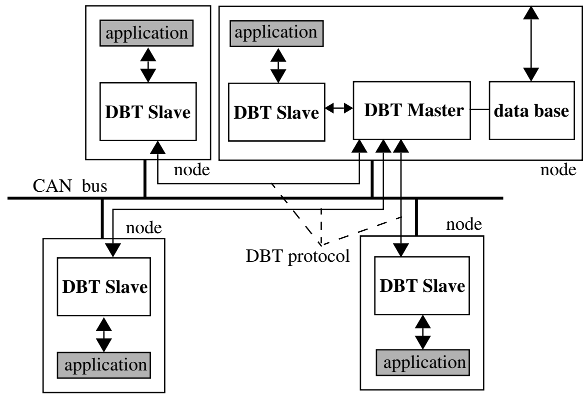

Fig. 6: distributor architecture

architectural issues. Each node contains a DBT slave, which allows the application process (server or client) to request an identifier for each object in use. Local objects are identified by predefined logical names. A central DBT master can look up object names for object identifiers. This assumes that the DBT master keeps a data base where a client server relation is defined via local object names. Figure 7 illustrates the protocol for identifier assignment. Note that additional parameters are exchanged, which we will deal with in Section 4.

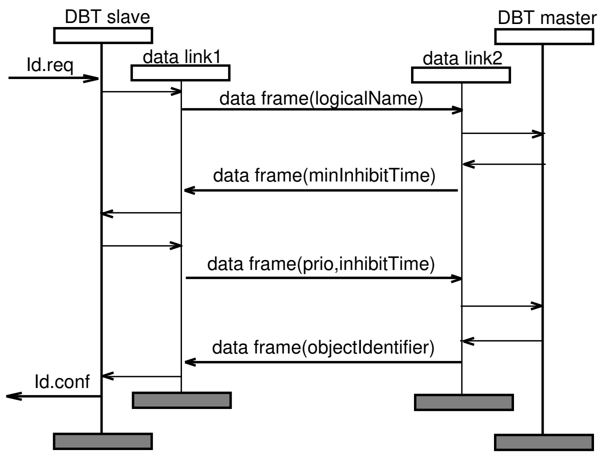

Fig. 7: distributor protocol

4. RESOURCE MANAGEMENT FOR CAN

In principle CAN applications can be implemented right on top of the data link layer ignoring CAL services. Thus a QoS enhanced communication service may be established either on the data link layer or on CAL. In both cases data/remote frames that carry the same object identifier obviously belong to the same flow. This notion of a flow differs from the traditional meaning: in current QoS architectures a flow denotes an unidirectional sequence of data packets that belong to the same connection. A CAN flow, however, may represent bidirectional interaction patterns consisting of data and remote frames, where interaction patterns follow from the particular object semantics. It is not a QoS specific task to conceive adequate object semantics, which strongly depend on the application domain. Therefore we avoid additional effort in this direction by simply implementing the desired QoS interface on the basis of ordinary CAL objects, for which the interaction patterns are already defined (see, e.g., the read-only variable or the uncontrolled event of Section 3.3). In order to yield QoS enhanced CAL communication services we instantiate the generic QoS architecture presented in Section 2.

We assume that the host processor, i.e. the CPU on which the application processes and the protocol entities of CAL run, suffers no capacity shortage. As a consequence we can limit resource management to the CAN controller[1], which is assumed to be the only bottleneck in the system.

Each of the following subsections instantiates a separate component of the generic QoS architecture, while Figure 8 gives a complete overview. Note that the proposed QoS architecture for CAN contains only one queueing station, namely the CAN bus itself, which, however, is utilized and scheduled in a distributed way. As a consequence, admission control can be centralized (located on a particular master node) while traffic policing for the only resource is performed at various places (namely at each network access point, not shown in Figure 8).

4.1 Service model

While retaining the object oriented service model of CAL, we propose a QoS enhanced service model, which supports object access with guaranteed QoS. Currently our service model supports only one service category, namely *bounded object access with deterministic delay.* According to Section 2.2 this service is characterized by a traffic model, the supported QoS parameters, and the obligation of guarantees:

- The traffic load is modelled as a *bounded* sequence of service requests. The arrival pattern of a sequence of service requests is bounded, if the next request cannot occur until a given amount of time (called interarrival time) elapses after the previous request. When an application process registers with the system, it has to specify an interarrival time for each CAL object it will use.
- If an application process commits to the specified interarrival time, the service provider guarantees that each object access will be finished before a predefined deadline expires. The application process has to specify the deadline for each of its CAL objects, when registering with the system. Note that for each CAL object the deadline must be less or equal to the interarrival time.

1. Note that CAN controllers implement the data link protocols (Sections 3.1, 3.2).

We basically assume that network nodes (including application processes) register with the system in the normal way, i.e. by the means of CAL network management services and the DBT protocol for assignment of object identifiers. The only exceptions are that the DBT protocol data units carry additional QoS parameters (interarrival time and deadline) and that the DBT master additionally performs an admission control test (see Sections 4.4 and 4.5).

As already mentioned, a service request may initiate a particular interaction pattern of CAN frames where the actual pattern depends on the semantics of the object that is accessed (see Section 3.3 for examples). Note that the completion of the whole interaction pattern must be guaranteed before the user specified deadline expires.

4.2 Scheduling discipline

It is straightforward to conclude the following from Section 3.2: if we assume that a binary 0 and 1 are coded as a dominant and recessive signal level, respectively, CAN frames with lower valued object identifiers precede frames with higher valued identifiers, when transmitted simultaneously. In this way the CAN medium access control implements a global fixed-priority scheduling of CAN frames, where preemption of ongoing transmissions is not allowed. Fixed-priority scheduling is well studied in the literature so that we can apply existing results to our proposed QoS architecture.

To complete the scheduling discipline we have to determine how priorities are assigned to CAN frames. As object identifiers (and therefore CAN frames) directly correspond to CAL objects we can derive exact knowledge about arrival patterns of CAN frames. As an example consider the access to a read-only variable as illustrated in Figure 4. It follows that each read request implies one remote frame and a subsequent data frame. That is, a read request triggers a sequence of two CAN frames until the read access is successfully completed, and there is no other situation where CAN frames with the same identifier are generated. Thus we can deduce that CAN frames of a particular CAL read-only variable occur in sequences whose starting frames lie at least the user specified interarrival time apart. It can be shown by analogy that CAN frames of a particular CAL uncon-

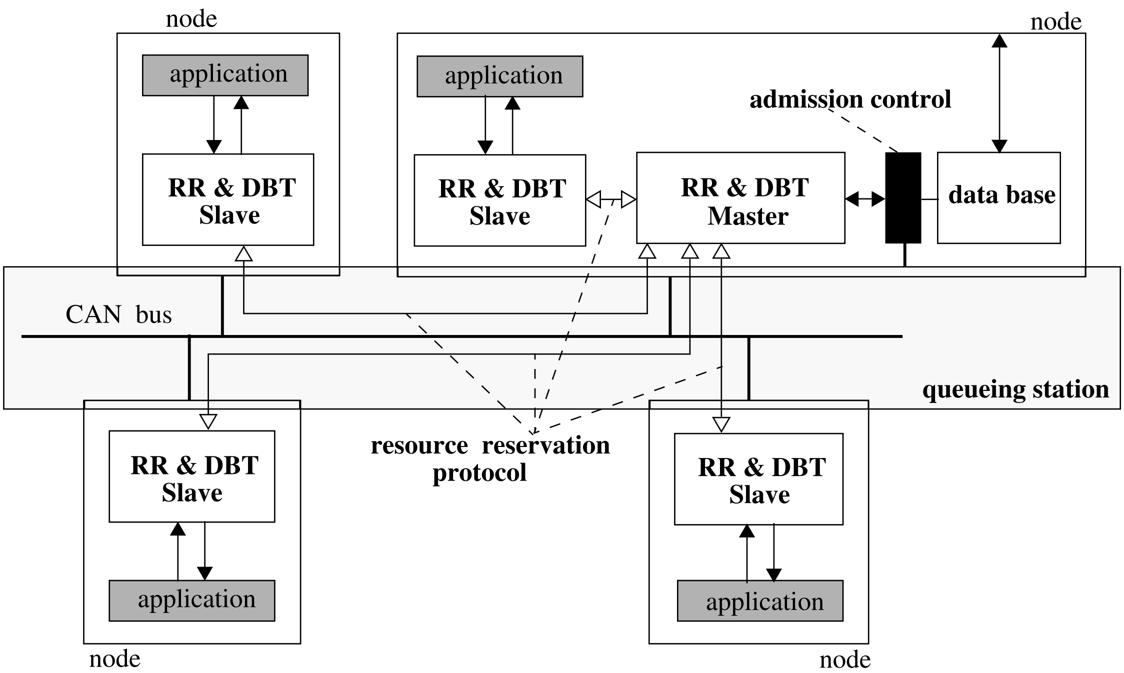

Fig. 8: QoS architecture for CAN

trolled event (Figure 5) occur in sequences of only one data frame that lie at least the user specified interarrival time (that corresponds to the uncontrolled event) apart. We call a sequence of CAN frames, that belongs to a specific service request, a transaction. Note that transactions of some enhanced CAL objects have two associated object identifiers. We propose to assign successive identifiers to objects that need more than one identifier and - in addition - to employ *deadline monotonic priority assignment*, i.e. transactions with shorter deadlines get higher priorities than transactions with longer deadlines. For deadline monotonic priority assignment Leung and Whitehead have shown the optimality of this strategy under fixed priority scheduling with deadlines at or before the end of the period [19]. This priority assignment strategy, however, has some unpleasant consequences:

- When priorities are assigned dynamically during system operation, it is possible that we run short of low priorities though there are yet higher priorities left, or we run short of high priorities though there are yet lower priorities left. To enforce deadline monotonic priority assignment in this case, it would be necessary to reconfigure priority assignment. Unfortunately, this causes a lot of overhead, when ongoing data transfers must not be disturbed. Alternatively, priority assignment could be refused, which also implies that the corresponding connection may be needlessly refused. As a compromise we relax deadline monotonic priority assignment and principally allow the assignment of another priority in case of shortage (as far as admission control passes).
- As priority reconfiguration is not supported we should apply a proper heuristic, when there are several choices to assign priorities. This may avoid shortage of priorities later on. For this purpose we suggest to divide the range of available priorities (1 to 1760 when CAL is supported) into segments that are to be associated with a particular interval of deadline values. Additionally it has to be ensured that each possible deadline corresponds to exactly one priority segment and that deadline intervals are associated monotonically to priority segments, i.e. shorter deadlines get higher priorities. In normal mode an object is assigned an arbitrary priority out of the segment that is associated with the specified object deadline. When a shortage of priorities occurs, the highest possible (according to admission control) priority from lower priority segments is selected. If not existent, the lowest possible priority from higher priority segments is selected. If this search fails again, the connection is refused.

4.3 Traffic policing

Application processes are not allowed to issue service requests with a higher frequency than agreed on. If a particular traffic contract is violated, the service request must be delayed until the specified interarrival time has been completely elapsed.

The CAL standard already supports so-called inhibit times (with the same meaning as interarrival times) to prevent high priority applications from consuming too much bandwidth. Inhibit times are assigned during execution of the DBT protocol (Figure 7). While DBT prescribes minimum inhibit times (set by the system operator in advance) to applications, our approach is different. The proposed QoS architecture for CAN allows the applications to dynamically specify a minimum interarrival time according to their needs while admission control checks for the availability of sufficient resources. This causes slight changes of the DBT protocol (Section 4.5).

4.4 Admission control

The overall network traffic load at some point in time during system operation characterizes as follows:

- There are k CAL objects in use where each object o_i is specified by an interarrival time T_i, a deadline D_i, and a transaction tr_i.
- A transaction tr_i itself is specified by the number n_i of involved CAN frames and the total length l_i of all involved CAN frames in bit.

In order to check for the availability of sufficient resources to meet all specified QoS parameters w.r.t. the expected traffic load and the employed scheduling discipline, a so-called schedulability test is needed. Such a test needs to assert a sufficient condition for schedulability, i.e. if a particular traffic load situation passes the schedulability test, QoS guarantees are met even under worst-case assumptions.

We applied certain results from scheduling theory [16] [17] [18] for the development of the following schedulability test.

Proposition 4.1 : *Consider a set of CAL objects o_1, o_2, ..., o_k arranged in priority order, i.e. lower indexed objects have higher priority. If $D_i \leq T_i$, then o_i will meet all its deadlines if*

$$\min_{0 < t \leq D_i} \sum_{j=1}^{i} \frac{l_j * b}{t} \left\lceil \frac{t}{T_j} \right\rceil + \frac{C_i}{t} + \frac{n_i * l_{max} * b}{t} \leq 1$$

where b denotes the bit time, i.e. the time necessary to send one bit on the medium, and l_{max} denotes the maximum length of a CAN frame. C_i denotes the time necessary to process the service request in the host processor.

If each CAL object in use passes this schedulability test, the whole set of CAL objects meets its deadlines. Thus if a new object is to be established, admission control selects a priority according to the procedure described in Section 4.2 and performs the above test for all objects with lower or equal priority. If the tests succeed, the new object will be accepted and refused otherwise.

4.5 Resource reservation protocol

CAN builds the trivial case of a packet-switched network where no intermediate systems exist. Thus the resource reservation protocol does not need to fix a route. Nevertheless, traffic loads are to be announced to the system, and the availability of sufficient network resources must be checked prior to data transfer. Furthermore a mechanism is needed to connect application processes.

CAN application processes are connected, if they share a common CAL object. These connections are established via the DBT protocol (Section 3.3). Apart from slight changes we reuse the DBT protocol as a resource reservation protocol (Figure 8). The combined resource reservation and distributor (RR & DBT) master resides on the node where admission control is located.

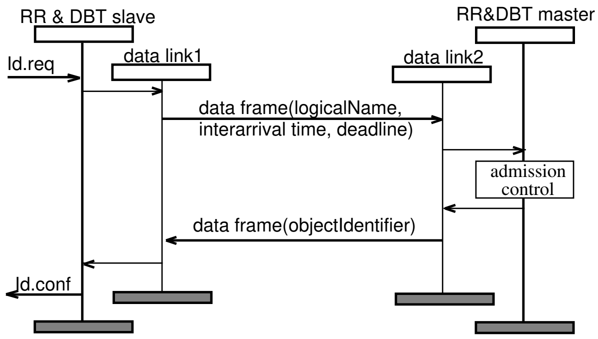

Fig. 9: resource reservation protocol

Figure 9 shows the details of the resource reservation protocol. Compared to the original distributor protocol the resource reservation protocol deletes the additional handshake for the assignment of the inhibit time and sends the QoS parameters together with the logical object name. After reception of the connection parameters the RR & DBT master checks for the availability of sufficient resources and - on success - sends back a proper object identifier to the RR & DBT slave.

Note that the reservation protocol also ignores the prio(rity) parameter, formerly offered by the DBT slave. This parameter specified the importance of the involved CAL object and was to be interpreted as a hint for a proper assignment of object identifiers. As admission control determines priority assignment solely on the basis of QoS and traffic parameters this is no longer necessary.

5. SUMMARY

We have presented a QoS enhancement for the CAN application layer CAL, which is based on a generic QoS architecture extracted from current QoS mechanisms proposed in the literature. We demonstrated that fieldbusses can benefit from the QoS architectures developed for Local and Wide area networks especially from dynamic resource management.

The potential for applying resource management to CAN is by no means exhausted. For instance, dynamic reconfiguration of existing priority assignments, the incorporation of additional service categories (e.g., aperiodic service requests) with advanced traffic models, or resource management for interconnected CAN busses are interesting fields for future research.

Acknowledgements. Special thanks go to our colleagues of the Sonderforschungsbereich SFB 501 for inspiring exchange of ideas.

REFERENCES

[1] K. Bender (ed.), PROFIBUS - Der Feldbus für die Automation (in german). 2. Auflage, Hanser Verlag, 1992

[2] A. Campbell, C. Aurrecoechea, and L. Hauw, "A Review of QoS Architectures", Proceedings of the 4th International IFIP Workshop on Quality of Service, 1996

[3] CAN in Automation (CiA) e.V., "Controller Area Network CAN - Ein serielles Bussystem nicht nur für Kraftfahrzeuge" (in german), 1996

[4] D. D. Clark, S. Shenker und L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", SIGCOMM'92 Symposium, 1992, pp. 14-26

[5] L. Delgrossi, L. Berger (ed.), "Internet Stream Protocol Version 2 (ST2) - Protocol Specification - Version ST2+", RFC 1819, 1995

[6] DIN 19245 Teil 1, "PROFIBUS Process Field Bus, Übertragungstechnik, Buszugriffs- und Übertragungsprotokoll, Dienstschnittstelle zur Anwendungsschicht, Management" (in german), 1991

[7] K. Etschberger, CAN Controller-Area-Network, Grundlagen, Protokolle, Bausteine, Anwendungen (in german). Hanser Verlag, 1994

[8] D. Ferrari, "Client Requirements for Real-Time Communication Services", IEEE Communications Magazine, 28(11), pp. 65--72, 1990

[9] D. Ferrari und D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", IEEE Journal on Selected Areas in Communications, 8(3), 1990, pp. 368-379

[10] H. Gerlach, "Neue Steuerungstechnik in der Elektroinstallation" (in german), etz, Bd. 110, Heft 22, 1989

[11] S. J. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management", SIGCOMM'90 Symposium, Communications Architecture & Protocols, 1990, pp. 8-18

[12] ISO 11898, "Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high speed communication"

[13] ISO 11519-2, "Road vehicles - Low speed serial data communication, part 2: Low speed Controller Area Network (CAN)"

[14] S. Jamin, S. Shenker, L. Zhang, and D. D. Clark, "An Admission Control Algorithm for Predictive Real-Time Service" (Extended Abstract), 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, 1992

[15] C. R. Kalmanek, H. Kanakia und S. Keshav, "Rate Controlled Servers for Very High-Speed Networks" IEEE Global Telecommunications Conference, 1990

[16] D. I. Katcher, S. S. Sathaye, and J. K. Strosnider, "Fixed Priority Scheduling with Limited Priority Levels", IEEE Transactions on Computers

[17] M.H. Klein, T. Ralya, B. Pollak, R. Obenza, M. G. Harbour, A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems. Kluwer Academic Publishers, 1993

[18] J. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization And Average Case Behavior", Real-Time Systems Symposium, 1989

[19] J. L. T. Leung and J. Whitehead, "On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks", Performance Evaluation (Netherlands), 2(4), 1982

[20] C. L. Liu und J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", Journal of the ACM, Vol. 20, No. 1, Jan. 1973, pp. 46-61

[21] A. K. Parekh und R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks - the Single Node Case", INFOCOM'92, 1992

[22] A. K. Parekh und R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks - the Multiple Node Case", INFOCOM'93, 1993

[23] D. Verma, H. Zhang und D. Ferrari, "Guaranteeing Delay Jitter Bounds in Packet Switching Networks", Tricomm'91, 1991

[24] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks", ACM Transactions on Computer Systems, Vol. 9, No 2, 1991, pp. 101-124

[25] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new Resource ReSerVation Protocol", IEEE Network, September 1993

[26] H. Zhang and D. Ferrari, "Rate-Controlled Static-Priority Queueing", Technical Report TR-92-003, Computer Science Devision, UCB, 1992

[27] H. Zhang und E. W. Knightly, "RCSP and Stop-and-Go: A Comparison of Two Non-Work-Conserving Disciplines for Supporting Multimedia Communication", Multimedia Systems Journal, September '95